A note on single-machine scheduling problems with position-dependent processing times

Julien Moncel, Gerd Finke, Vincent Jost

A note on single-machine scheduling problems with position-dependent processing times

Julien Moncel[*] Gerd Finke[†] Vincent Jost[‡]

Abstract

The purpose of this note is two-fold. First, it answers an open problem about a single-machine scheduling problem with exponential position-dependent processing times defined in [V. S. Gordon, C. N. Potts, V. A. Strusevich, J. D. Whitehead, *Single machine scheduling models with deterioration and learning: Handling precedence constraints via priority generation*, Journal of Scheduling **11** (2008), 357–370]. In this problem, the processing time of job i when scheduled in rank r is equal to $p(i,r) = p_i\gamma^{r-1}$, with γ a positive constant. Gordon *et al* show in the above-mentioned paper with priority-generating techniques that the problem of minimizing the total flow-time on one machine admits an $O(n \log n)$ algorithm when $\gamma \in]0,1[\cup [2,+\infty[$, and leave the case $\gamma \in [1,2[$ open. We show that the problem admits an $O(n \log n)$ algorithm also for $\gamma \in [1,2[$. The second purpose of this note is to provide a simple and general insight on why and when position-dependent scheduling problems on one machine can be solved in time $O(n \log n)$.

Corresponding author : Julien Moncel `julien.moncel@iut-rodez.fr`
Phone: +33(0)5 65 77 10 80 – Fax: +33(0)5 65 77 10 81

Keywords : Single-machine scheduling ; position-dependent processing times ; learning effects ; deteriorating jobs

1 Scheduling with position-dependent processing times

There is a growing literature dealing with scheduling problems where the actual processing time of a job depends on its position in the schedule, and/or its starting processing time (see for instance the recent monography [4] for a survey on time-dependent scheduling). This enables in particular one to model the so-called learning and deteriorating effects. Practical applications include operators becoming more efficient while getting used to a new procedure (learning effect), and forest fires that take longer to extinct as time flows (deteriorating effect).

[*]CNRS – LAAS Université de Toulouse, UPS, INSA, INP, ISAE ; UT1, UTM, LAAS, 7 avenue du Colonel Roche, 31077 Toulouse Cédex 4 (France), also with Fédération de recherche Maths à Modeler, and also with Université Toulouse 1 Capitole, IUT Rodez

[†]Laboratoire G-SCOP, 46 avenue Félix Viallet, 38031 Grenoble Cédex (France)

[‡]École polytechnique, Laboratoire d'informatique (LIX), 91128 Palaiseau Cédex (France)

In this paper we focus on position-dependent processing times, where the processing time of job i when scheduled in rank r is equal to $p(i,r)$. Hence processing times of jobs are defined by a function $p : i, r \mapsto p(i,r)$. For a general function p it is known that problems $1 \mid p(i,r) \mid C_{\max}$ and $1 \mid p(i,r) \mid \sum C_i$ can be modelled as assignment problems and thus admit $O(n^3)$ algorithms [1, 2] (n being the number of jobs).

But for most practical purposes we do not need processing times in such a general form as $p(i,r)$, and the problem can be solved with simpler methods than solving an assignment problem. Let us assume that the function $p(i,r)$ can be written as $p(i,r) = f(r)p_i$. In this case, we say that the position-dependent scheduling times are *decomposable*, p_i being the *normal* processing time of job i, and $p(i,r)$ its *actual* processing time if scheduled in position r. This is the case for many scheduling problems of the literature, such as the model of Biskup [2] $p(i,r) = p_i r^a$ (with $a < 0$ a constant "learning index"), the model of Wang and Xia [11] $p(i,r) = p_i(a - br)$ (with $a \geq 0$ integer, $b \geq 0$ rational, and $a-(n+1)b > 0$), or the model of Gordon *et al* [5] $p(i,r) = p_i\gamma^{r-1}$ (with $\gamma > 0$). For this last model, it is shown with priority-generating techniques in [5] that the problem $1 \mid p(i,r) = p_i\gamma^{r-1} \mid \sum C_i$ can be solved in time $O(n \log n)$ for $\gamma \in]0,1[\cup [2,+\infty[$, the case $\gamma \in [1,2[$ being left open. In the next section we prove that this last problem admits an $O(n \log n)$ algorithm for every $\gamma > 0$.

We get this result as a consequence of a more general result on scheduling jobs on one machine with position-dependent processing times. Let us say that an objective function γ is *decomposable* if it can be written as $\gamma = \sum \nu_r p_{[r]}$, where $p_{[r]}$ denotes the actual processing time of job scheduled in position r, and $\nu_1, \ldots, \nu_n$ are parameters that depend on the number of jobs of the problem but not on the processing time of the jobs. Many classical objective functions are decomposable. Trivially, $C_{\max} = \sum p_{[r]}$ is decomposable (we have $\nu_r = 1$ for all r). Similarly, since $\sum C_i$ can be rewritten as $\sum (n+1-r)p_{[r]}$, then it is a decomposable objective function with $\nu_r = (n+1-r)$ for all r. Other functions are decomposable, such as for instance the total absolute differences in completion times (TADC), defined as $\mathrm{TADC} = \sum_{i<j} |C_i - C_j|$. Indeed, it is easy to see that TADC can be rewritten as $\sum \nu_r p_{[r]}$ with $\nu_r = \sum_{j \geq r}(2j-(n+1))$ for all r.

In the next section, we show that, if both the objective function γ and the scheduling times $p(i,r)$ are decomposable, then the single-machine scheduling problem $1 \mid p(i,r) \mid \gamma$ admits an $O(n \log n)$ algorithm, that consists essentially in sorting two series of numbers. Some consequences of this result are discussed in Section 3. We then provide in Section 4 a characterization of the processing times for which an optimal schedule can be found by a sorting algorithm.

2 A general result on decomposable objective functions and position-dependent processing-times

We start by a well-known lemma of Hardy *et al* [6] on minimizing the scalar product of the permutation of two sequences of numbers.

Lemma 1 (Hardy *et al*) *Let $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$ be two sequences of numbers, and let us assume that $x_1 \leq x_2 \leq \ldots \leq x_n$. Let π denote a per-*

mutation on $\{1, \ldots, n\}$. Then the the minimum of $\sum x_i y_{\pi(i)}$, taken over all permutations of $\{1, \ldots, n\}$, is attained for any π^* satisfying $y_{\pi^*(1)} \geq y_{\pi^*(2)} \geq \ldots \geq y_{\pi^*(n)}$. $\square$

The proof of this lemma is easy, since it suffices to notice that if $x_1 \leq x_2$ and $y_1 \leq y_2$, then $x_1 y_2 + x_2 y_1 \leq x_1 y_1 + x_2 y_2$. The next theorems are consequences of this result.

Theorem 1 *Let γ be a decomposable objective function. Then any single-machine scheduling problem $1 \mid p(i,r) \mid \gamma$ can be modelled by an assignment problem, and thus solved in time $O(n^3)$, where n is the number of jobs.*

Proof: The result is immediate. Indeed, by definition, if γ is decomposable, then it can be written as $\gamma = \sum \nu_r p_{[r]}$, where $p_{[r]}$ denotes the (actual) processing time of job scheduled in position r. This can be seen as an assignment problem, where the weight from job i to position r is $\nu_r p_i$. $\square$

If the processing times are also decomposable then we get the following stronger result.

Theorem 2 *Let γ be a decomposable objective function, and let us assume that the position-dependent processing times of jobs are also decomposable. Then any single-machine scheduling problem $1 \mid p(i,r) = f(r)p_i \mid \gamma$ can be solved in time $O(n \log n)$, where n is the number of jobs.*

Proof: By definition, if γ is decomposable, then it can be written as $\gamma = \sum \nu_r p_{[r]}$, where $p_{[r]}$ denotes the (actual) processing time of job scheduled in position r. Let us assume that the schedule is described by a permutation π, such that $\pi(r) = i$ if and only if job i is scheduled in position r. Now, we clearly have $p_{[r]} = p(\pi(r), r) = f(r)p_{\pi(r)}$, such that $\gamma = \sum \nu_r p_{[r]} = \sum \nu_r f(r) p_{\pi(r)}$. By Lemma 1, it suffices to sort the parameters $\nu_r f(r)$ in non-decreasing order, and sort the jobs in non-increasing order of their normal processing times in order to minimize the objective function γ. To terminate the proof it then suffices to notice that sorting two sequences of n numbers can be made in time $O(n \log n)$. $\square$

This last theorem shows that, if both the objective function and the processing times are decomposable, an optimal schedule can be found by running a sorting algorithm. Indeed, assuming that the jobs are sorted in an SPT order (that is to say $p_1 \leq p_2 \leq \ldots \leq p_n$), there exists a fixed permutation π (that depends only on the function f and on γ) such that the schedule defined by "i is scheduled at rank r if and only if $\pi(r) = i$" is optimal. This permutation π is defined by $\nu_{\pi^{-1}(1)} f(\pi^{-1}(1)) \geq \nu_{\pi^{-1}(2)} f(\pi^{-1}(2)) \geq \ldots \geq \nu_{\pi^{-1}(n)} f(\pi^{-1}(n))$.

3 Some consequences of the general result in the decomposable case

Theorem 2 generalizes and unifies in a single framework many results of the literature, including those described in Table 1.

Reference	Problem
[2]	$1 \mid p(i,r) = p_i r^a \mid \sum C_i$ (with $a < 0$)
[7]	$1 \mid p(i,r) = p_i r^a \mid C_{\max}$ (with $a < 0$)
[8]	$1 \mid p(i,r) = p_i r^a \mid C_{\max}$ (with $a > 0$)
[10]	$1 \mid p(i,r) = p_i(M + (1-M)r^a) \mid C_{\max}$ (with $a \leq 0$ and $M \in [0,1]$)
[11]	$1 \mid p(i,r) = p_i(a - br) \mid \sum C_i$ and $1 \mid p(i,r) = p_i(a - br) \mid C_{\max}$ (with $a \geq 0$ integer, $b \geq 0$ rational, and $a - (n+1)b > 0$)
[5]	$1 \mid p(i,r) = p_i \gamma^{r-1} \mid \sum C_i$ (with $\gamma \in]0,1[\cup [2,+\infty[$)
[12]	$1 \mid p(i,r) = p_i r^a \mid$ TADC (with $a < 0$)
[9]	$1 \mid p(i,r) = f(r)p_i \mid C_{\max}$ (with f increasing or decreasing)

Table 1: Sample of existing results of the literature that are generalized by Theorem 2. These results are sorted chronologically. Most of them use an interchange argument, and some explicitly use Lemma 1 (for instance [12] and [9]).

Mosheiov shows in [8] that there always exists a so-called "V-shaped" optimal schedule for problem $1 \mid p(i,r) = p_i r^a \mid \sum C_i$ (with $a > 0$). Recall that a schedule is said "V-shaped" if it consists of a subset of jobs arranged in a non-increasing order of processing times, followed by the remaining jobs in non-decreasing order of their processing times. This result can also be seen as a consequence of Theorem 2. Indeed, in this case we have $\sum C_i = \sum (n+1-r)p_{[r]} = \sum (n+1-r) r^a p_{[r]}$. The result follows from the fact that $g : r \mapsto g(r) = (n+1-r)r^a$ is $\wedge$-shaped (that is to say it is impossible to have simultaneously $g(r) < g(r-1)$ and $g(r) < g(r+1)$ for $1 < r < n$).

Whereas most of the results listed in Table 1 use an interchange argument, the result of Gordon *et al* [5] is based on priority-generating techniques and is rather involved. This in particular explains why they get a proof only for $\gamma \in]0,1[\cup [2,+\infty[$. Indeed, they show that for $\gamma \in]1,2[$ there does not exist 1-priority functions for the problem (we refer the interested reader to [5] for the definition of priority functions). As an immediate consequence of Theorem 2, we get the following.

Corollary 1 *The problem* $1 \mid p(i,r) = p_i \gamma^{r-1} \mid \sum C_i$ *(with* $\gamma > 0$*) admits an* $O(n \log n)$ *algorithm.* □

4 Characterisation of sortable processing times

Let us consider a decomposable objective function $\gamma = \sum \nu_r p_{[r]}$. We now show that decomposability of processing times is not necessary for the existence of a fixed permutation yielding an optimal schedule provided the processing times are sorted. Let $\mathcal{P} \subseteq \mathbb{R}_+$ be the set of all the possible normal processing times of the jobs, and let us assume now that $p(i,r)$ can be seen as $f_r(p_i)$. In this framework, each f_r is a function $f_r : \mathcal{P} \to \mathbb{R}_+$. For all r, set $g_r = \nu_r f_r$, and define $\mathcal{G}$ as the set $\{g_1, \ldots, g_n\}$. We say that $g_r \succeq g_s$ if

$$g_r(p) - g_r(q) \geq g_s(p) - g_s(q) \qquad \forall (p,q) \in \mathcal{P}^2 \text{ with } p \geq q \tag{1}$$

Clearly, $\succeq$ defines a preorder on $\mathcal{G}$ (that is to say, $\succeq$ is reflexive and transitive). We say that g_r and g_s are *comparable* if either $g_r \succeq g_s$ or $g_s \succeq g_r$.

The justification of this definition of comparability of processing times is two-fold. One is Theorem 3 below stating somehow that comparability is the essential property for the double-sorting algorithm of Theorem 2 to work. The other is the variety of examples of comparable processing times, including for instance if $\gamma = C_{\max}$:

- $f_r(p_i) := k_r p_i$ for $\mathcal{P} \subseteq \mathbb{R}_+$ and $k_r \in \mathbb{R}_+$
- $f_r(p_i) := p_i^{k_r}$ for $\mathcal{P} \subseteq [1, +\infty[$ and $k_r \in \mathbb{R}_+$
- $f_r(p_i) := k_r^{p_i}$ for $\mathcal{P} \subseteq \mathbb{R}_+$ and $k_r \in \mathbb{R}_+$

Note also that $f_r \succeq f_s$ and $f'_r \succeq f'_s$ implies $f_r + f'_r \succeq f_s + f'_s$.

The following theorem states that, provided the processing times are sorted in an SPT order, there exists a fixed permutation yielding an optimal schedule if and only if the functions g_r are all pairwise comparable.

Theorem 3 *Let $\gamma = \sum \nu_r p_{[r]}$ be a decomposable objective function. Let n be any number of jobs, $\mathcal{P}$ be any set of admissible processing times, and $f_1, \ldots, f_n$ be n functions from $\mathcal{P}$ to $\mathbb{R}_+$. Let $\mathcal{G} = \{g_r \mid 1 \leq r \leq n\}$, where for all r the function g_r is defined as $\nu_r f_r$. Then $(\mathcal{G}, \succeq)$ is a totally preordered set if and only if there exists a permutation π on $\{1, \ldots, n\}$ such that, for any instance $(p_1, \ldots, p_n) \in \mathcal{P}^n$ of $1 \mid f_r(p_i) \mid \gamma$ such that $p_1 \leq p_2 \leq \ldots \leq p_n$, assigning job i to rank r if and only if $\pi(r) = i$ leads to an optimal schedule.*

Proof: Let us first assume that $(\mathcal{G}, \succeq)$ is a totally preordered set, that is to say the functions g_r are all pairwise comparable. In this case, there exists a permutation π such that $g_{\pi^{-1}(1)} \succeq g_{\pi^{-1}(2)} \succeq \ldots \succeq g_{\pi^{-1}(n)}$. Consider now any instance $(p_1, \ldots, p_n) \in \mathcal{P}^n$ of $1 \mid f_r(p_i) \mid \gamma$ such that $p_1 \leq p_2 \leq \ldots \leq p_n$. Then an interchange argument shows that assigning job i to rank r if and only if $\pi(r) = i$ leads to an optimal schedule. Indeed, if two jobs i and j such that $p_i \leq p_j$ are scheduled i at a rank r, and j at a rank s, with $g_s \succeq g_r$, then exchanging jobs i and j can only improve γ.

Now let us assume that there exist two functions g_r and g_s that are not comparable. This implies that there exist $p \geq q$ and $p' \geq q'$ such that $g_r(p) - g_r(q) > g_s(p) - g_s(q)$ and $g_r(p') - g_r(q') < g_s(p') - g_s(q')$. As a consequence, there can not exist a fixed permutation π on $\{1, \ldots, n\}$ such that, for any instance $(p_1, \ldots, p_n) \in \mathcal{P}^n$ of $1 \mid f_r(p_i) \mid \gamma$ such that $p_1 \leq p_2 \leq \ldots \leq p_n$, assigning job i to rank r if and only if $\pi(r) = i$ leads to an optimal schedule. Indeed, let us consider one instance such that $p_1 = q$ and $p_2 = p$, and another instance such that $p_1 = q'$ and $p_2 = p'$. Since $g_r(p) - g_r(q) > g_s(p) - g_s(q)$ and $g_r(p') - g_r(q') < g_s(p') - g_s(q')$, then for one of these instances job 1 is scheduled before job 2 in all optimal schedules, and for the other one job 2 is scheduled before job 1 in all optimal schedules. $\square$

5 Conclusion

In this note we presented general results on decomposable objective functions and position-dependent processing-times. These results cover in particular the classical objective functions $C_{\max}$, $\sum C_i$, and TADC. They also generalize several existing results of the literature. In particular, Theorem 2 states that any problem of the form $1 \mid p(i,r) = f(r)p_i \mid \gamma$ can be optimally solved by a sorting algorithm if γ is decomposable. This theorem simplifies a result of Gordon *et al* on a single-machine scheduling problem with exponential position-dependent processing times [5], and enables one to extend this result.

Furthermore, Theorem 3 provides a characterization of processing times for which there exists a sorting algorithm that optimally solves any problem of the form $1 \mid p(i,r) = f_r(p_i) \mid \gamma$ in the case where γ is decomposable. This result uses a notion of comparability between functions, which in some sense is an extension of so-called Monge properties for matrices (see e.g the survey [3]).

References

[1] A. Bachman, A. Janiak, *Scheduling jobs with position-dependent processing times*, Journal of the Operational Research Society **55(3)** (2004), 257–264.

[2] D. Biskup, *Single-machine scheduling with learning considerations*, European Journal of Operational Research **115** (1999), 173–178.

[3] R. E. Burkard, B. Klinz, R. Rudolf, *Perspectives of Monge properties in optimization*, Discrete Applied Mathematics **70** (1996), 95–161.

[4] S. Gawiejnowicz, *Time-Dependent Scheduling*, Monographs in Theoretical Computer Science – An EATCS Series, Springer (2008).

[5] V. S. Gordon, C. N. Potts, V. A. Strusevich, J. D. Whitehead, *Single machine scheduling models with deterioration and learning: Handling precedence constraints via priority generation*, Journal of Scheduling **11** (2008), 357–370.

[6] G. Hardy, J. E. Littlewood, G. Pólya, *Inequalities*, 2nd edition, Cambridge University Press (1988).

[7] G. Mosheiov, *Scheduling problems with a learning effect*, European Journal of Operational Research **132** (2001), 687–693.

[8] G. Mosheiov, *A note on scheduling deteriorating jobs*, Mathematical and Computer Modelling **41** (2005), 883–886.

[9] K. Rustogi, V. A. Strusevich, *Single Machine Scheduling with General Positional Deterioration and Rate-Modifying Maintenance*, Technical Report SORG-03/2011, University of Greenwich (2011).

[10] J.-B. Wang, M.-Z. Wang, Z.-Q. Xia, *Single-machine scheduling with a general learning effect*, Journal of Mathematical Research and Exposition **25** (2005), 642–646.

[11] J.-B. Wang, Z.-Q. Xia, *Flow-shop scheduling with a learning effect*, Journal of the Operations Research Society **56** (2005), 1325–1330.

[12] D. L. Yang, W.-H. Kuo, *Some scheduling problems with deteriorating jobs and learning effects*, Computers and Industrial Engineering **58** (2010), 25–28.